2015 IEEE Symposium Series on Computational Intelligence

Collaborative Filtering of Call for Papers

He-Da Wang* and Ji Wu†
Multimedia Signal and Intelligent Information Processing Laboratory
Department of Electronic Engineering
Tsinghua University, Beijing, China
Email: *wang-hd12@mails.tsinghua.edu.cn, †wuji_ee@mail.tsinghua.edu.cn

***Abstract*—Call for papers (CFPs) are notifications of academic events that invite researchers to submit their works. Traditionally, CFPs are handed out to researchers by mailing lists and web pages. With the number of conferences increasing, finding, reading and filtering out relevant CFPs become time consuming and need the assistance from information retrieval techniques. In this paper, we employ collaborative filtering to match relevant CFPs to researchers. Non-personalized, neighborhood-based and class-based methods are applied in CFP recommendation. We also propose a hybrid approach that utilizes conference series and submission deadlines of CFPs. The experiments on WikiCFP data set show that the class-based method outperforms both neighborhood-based and non-personalized methods, whereas the proposed hybrid approach has the best overall performance.**

I. Introduction

Conferences and workshops play an important role in academic communication. Invitations for calling scientists to submit their works to such events are often named call for papers (CFP). A typical CFP is a document including the location, time, submission deadline, and topics of the events. It is either published as a web page on dedicated conference websites or distributed by one-to-many emails and mailing lists. Reading them helps researchers to decide whether to submit their works to such events and to schedule their researches.

Today, increasing number of academic events contributes to a boom of CFPs. Researchers have to spend more time dealing with them by either going through the mailing lists or searching the web for relevant CFPs. Efforts have been made to lessen the burden for researchers by aggregating a lot of CFPs in one database and providing various kinds of information filtering service. Such solutions include RSS subscription resources (eg. CFP Website of Upenn[1]), mailing lists (eg. DBWorld[2]), and dedicated database websites (eg. CFPList[3], WikiCFP[4]).

Finding CFPs from such sources is still not a happy experience. There are an overwhelming number of CFPs now: WikiCFP alone has collected over 40,000 CFPs according to its statistics. Unless you have the name of the conference in mind, or you have to manually go through the list and find relevant ones. This is both difficult and time consuming, and facilitates the need for automatic information filtering methods to find CFPs relevant to one's personal interest.

[1]http://call-for-papers.sas.upenn.edu
[2]https://research.cs.wisc.edu/dbworld/
[3]http://www.cfplist.com
[4]http://www.wikicfp.com

Recommender systems have been employed in many real world applications to mitigate information overload. There are applications that find collaborators [1] and recommend new scientific articles [2] for researchers. It might also be useful if a personalized recommender can match researchers to relevant CFPs and save them a lot of effort.

Despite the pain in finding CFPs, scientific discussions about CFP recommendation are rare. Martín et al. [3] proposed a content-based recommender system of CFPs, which is, to our best knowledge, the only one trying to address this problem in recommender system literature. There are others trying to address this problem by improving CFP retrieval performance. Xia et al. [4] proposed the use of social tags in CFP classification to increase the accuracy. Issertial and Tsuji [5] showed that adding enriched ontology into structure data of CFPs will increase the recall in CFP retrieval.

So far, collaborative filtering of CFP has not yet been discussed in the literature of recommender system. This is partly because the absence of an available user preference data set. In this paper, we utilize an implicit feedback called the "track by" relation from a real-world data source in our recommender system. This kind of feedback allows us to employ collaborative filtering in CFP recommendation.

Our contributions to CFP recommendation are as follows: we introduce a novel CFP data set containing implicit feedbacks; we compare the performances of non-personalized, neighborhood-based and class-based collaborative filtering algorithms on this data set; we propose a hybrid approach that blends content information of the CFPs into collaborative filtering, which outperforms all other approaches.

The next section introduces the date set and the problem setting of this work. Section III reviews the previous works about collaborative filtering and CFP recommendation. Section IV introduces the algorithms we use in CFP recommendation. Section V explains our experiment settings. Section VI shows the results of the experiments. We conclude our work and suggest some open problems for future exploration in the last section.

II. Problem Settings

A. Dataset

We crawl open-access data from WikiCFP and use it in our experiments. WikiCFP is a crowd-source database that collects CFPs and provides retrieval services. One can search for CFPs by its keyword, add CFPs to one's own list and get email notifications before submission deadlines. A sample CFP data on WikiCFP is shown in Fig. 1.

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.264

Fig. 1. A typical Call-for-Paper on WikiCFP. The meta data consists of a summary, a description, a link to the conference website, when and where the event will be held, and several category labels added by the users who upload or modify it.

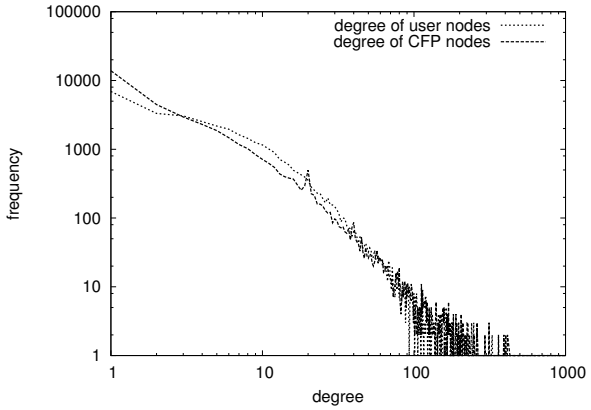

Fig. 2. Degree distribution of the bipartite graph consists of *tracked-by* relations follows an approxi-mate power law.

There are three ways to find a CFP on WikiCFP: searching for a keyword, browsing by category and browsing by conference series. If you find a CFP that is relevant to your need, you can *add* it to your list. Then, you can manage your own list of selected CFPs and browse the list in a time-line style user interface. You can also post a new CFP into the database and everyone will be able to see it after it is reviewed. On the page of a CFP, you can see which user uploads this CFP (displays "posted by") and who have added it to their own lists (displays "tracked by").

Such implicit feedbacks of *posted-by* and *tracked-by* relations provide us with the collaborative information in CFP managing. We extract 389,213 *tracked-by* relations from 35,725 CFP items to 38,431 users from the crawled pages. We also extract 29,542 *posted-by* relations from 29,542 CFP items to 7,842 users, which is less than the total count of CFPs since some of them are posted by system. As a CFP will be automatically tracked by the user who uploads it, we can only consider the *tracked-by* relations. The *tracked-by* relations between users and CFPs form a bipartite graph, the degree distribution of which is shown in Fig. 2.

It should be noted that the *tracked-by* relations in the data set have no temporal information about when the users track the CFPs. This makes a recommender system with temporal dynamics inapplicable.

B. Problem Definition

The data set we use consists of implicit feedbacks. Different from explicit feedbacks like 1-5 star ratings, which clearly convey intended user preference, implicit feedbacks are interpreted according to our understanding of the motivation of the user. That a conference is tracked by a user does not mean the user likes it or dislikes it. However, as the users depend on their lists to manage their tracked CFPs, we can reasonably assume that *tracked-by* is a sign of relevance.

We choose top-N recommender as the setting of the CFP recommender system. In this scenario, a recommender system RS uses feedback set R to learn user profiles. The feedback set R is a $|U| \times |I|$ binary matrix defined by *tracked-by* relations, in which U stands for the set containing all users and I stands for the set containing all CFP items.

$$r_{ui} = \begin{cases} 1 & \text{if CFP } i \text{ is tracked by user } u; \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

For every user $u \in U$, RS recommends the most relevant N CFPs to u. The performance of RS depends on the percent and rank of true relevant one in recommended items.

III. Related Works

A. Collaborative Filtering

The purpose of collaborative filtering is to recommend items with highest utility to the active user (the user that recommender system currently serves) by finding patterns from the preferences of all users. The approaches of collaborative filtering can be divided into two categories, the memory-based and model-based methods.

Memory-based methods keep track of all preferences from the users and combine them to make predictions. One of the memory-based approaches is the neighborhood-based approach [6], [7]. In a user-based neighborhood method, users who are the most similar to the active user are called nearest neighbors, and preferences of the nearest neighbors are used to predict the preferences of the active user. The counterpart is the item-based neighborhood method, which utilizes similarities between pairs of items. Interpretations of such methods using implicit feedbacks are discussed in both industrial [8] and academic [9] literatures.

Model-based methods learn a model from preferences and use it in prediction. There are many model-based methods proposed to address the collaborative recommendation problem. Among these models, graph-based model and latent class model are two widely discussed ones. Graph-based models such as ItemRank [10] and RWR [11] perform random walk with restart on the bipartite graph consists of users, items and relations between them. They are suitable for both implicit and explicit feedbacks. Latent class models are probabilistic graphic models that cluster users into latent classes [12]. The original latent class model is applicable to implicit feedbacks and it is also extended to a version that can use preference values.

B. CFP Recommendation

Martín et al. [3] study several content-based methods to match CFPs to the users. They concatenate the description of a CFP with the abstracts from the papers wrote by the people in the program committee of the corresponding conference to construct item profile for the CFP. They download all the papers wrote by the users and use the abstracts from these papers as the user profiles. A total of 13 researchers participate as the users to test various content-filtering approaches in their experiment. The relevance scores of CFPs are manually annotated by these 13 participants.

Despite the meaningful exploratory work of [3], we decide not to extend their work, as we notice several inconsistencies between our problem setting and theirs. The major problem is that a user profile in their setting consists of concatenation of previously published abstracts of the user, while our data set contains no identity information that allows us to find previous papers of the user. Also, the *tracked-by* relations in WikiCFP data set are not completely equivalent to relevance scores annotated by human participants. That a user on WikiCFP tracks a CFP means that the user considers the conference as a candidate event to submit her/his works to, while a "relevant" annotation only means the user thinks the conference's topic is relevant to her or his field of interest.

IV. Recommending Call For Papers

We employ four different strategies in our experiment of CFP recommendation: there are a non-personalized method, two neighborhood-based methods, and one based on latent class model. We also propose a novel hybrid approach that utilizes domain-specific properties of CFP: conference series and submission deadline.

A. Collaborative Filtering of CFPs

a) Popularity: As CFPs from well-known conferences will get major attention and attract a lot of researchers to track them, the number of being tracked reflects the popularity of a conference. Ranking CFPs by how many users track them is an intuitive way to do non-personalized recommendation.

This strategy is efficient for its brevity; it is also effective if popular items are in domination. However, from Fig. 2 we can tell that the data from WikiCFP have a "long tail", which means *tracked-by* relations from CFPs that are less tracked take up a major part. We could anticipate that popularity strategy will perform badly.

b) Neighborhood: As a user's research interest might be similar to those who track the same CFPs, the user might share common interest in other CFPs with these like-minded users. Therefore, if we find these like-minded users and let them vote for their most interested CFPs, we might find relevant CFPs in their voting items. This is the original intuitive of neighborhood-based collaborative filtering [6]. In this approach, users most similar to the active user are called nearest neighbors. The CFPs most tracked by these nearest neighbors are treated as their votes. The most voted ones are recommended to the user, weighted by the degree of like-mindedness between the neighbor and the user:

$$\hat{r}_{u,i} = \sum_{v \in NN} Sim(u,v) r_{v,i} \tag{2}$$

in which similarity $Sim(u,v)$ represents the like-mindedness between the user u and the neighbor v. It acts just as the vote weight in shareholder meetings. The more alike a neighbor is to the user, the more weight the vote carries.

We use Jaccard index [13], [14] here to measure the similarity between two users. This coefficient measures the similarity between two set of items. We use it to measure the percent of overlapped items in items tracked by one user and items tracked by the other user:

$$Jaccard(u,v) = \frac{|R_u \cap R_v|}{|R_u \cup R_v|} \tag{3}$$

in which R_u is the set containing all CFP items that user u tracks:

$$R_u = \{i \in I | r_{ui} = 1\} \tag{4}$$

c) Neighborhood-IDF: The Jaccard index between two users reflects how many of their interested events are commonly interested, which means the higher the percent of commonly tracked events, the higher their resemblance according to the similarity measure.

However, this measure fails to reflect the difference between items. Comparing to CFPs tracked by most of the users, sharing less tracked CFPs indicates more resemblance. For example, "IJCAI" is almost tracked by everyone. On the other hand, "RecSys" is relatively less tracked by. Then it is reasonable to assume that the similarity between two users who track "RecSys" is stronger than that between two users tracking "IJCAI".

We employ inverse document frequency (IDF) to highlight the importance of less popular CFPs. IDF is first introduced as a term weighting technique by [15] which notes "...terms should be weighted according to collection frequency, so that matches on less frequent, more specific terms are of greater value...".

The IDF of a CFP i is defined as an inverse function of the frequency of being tracked by the users:

$$IDF(i) = log(\frac{|U|}{|R_i|}) \tag{5}$$

in which R_i is the set of users who track CFP i:

$$R_i = \{u \in U | r_{ui} = 1\} \tag{6}$$

We integrate IDF into Jaccard index by applying weighting factors to CFPs:

$$Jaccard_{idf}(u,v) = \frac{\sum_{i \in R_u \cap R_v} log(\frac{|U|}{|R_i|})}{\sum_{i \in R_u \cup R_v} log(\frac{|U|}{|R_i|})} \tag{7}$$

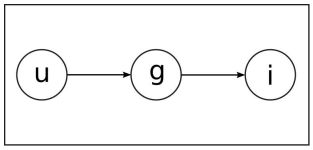

Fig. 3. Graphical model representation of latent class model.

In this way, we stress the importance of CFPs from more specific conferences in the computation of user similarity. To our best knowledge, we are the first to refine the Jaccard index with IDF weighting. We denote this revised approach by *Neighborhood-IDF*.

d) Latent Class Model: It is straightforward to select like-minded users and use their votes in recommendation. Whereas, the neighborhood can be small if the user has only tracked a few CFPs. This would lead to bad quality of recommendation.

Clustering approaches address this issue by dividing users into clusters, in a way that two users in one cluster are alike and two users in different clusters are not. Latent class model [12] does similar things besides it assigns more than one cluster to each user, but considers each user as a probabilistic mixture of representative groups. According to the latent class model, each user u belongs to a group $g \in G = \{g_1, g_2, ..., g_K\}$ by probability $P(g|u)$ and each group g considers a CFP i relevant by probability $P(i|g)$. Then the probability that a user u tracks a CFP i is:

$$P(i|u) = \sum_{g \in G} P(i|g)P(g|u) \tag{8}$$

which is derived from the key assumption of latent class model [12] that u and i are independent conditioned on g:

$$P(u,i) = \sum_{g} P(u,i|g)P(g) = \sum_{g} P(i|g)P(u|g)P(g) \tag{9}$$

Items with the highest possibility of being tracked $P(i|u)$ are recommended to user u based on the latent class model. The model parameter $P(i|g)$ and $P(g|u)$ can be estimated by Expectation Maximization (EM) algorithm [16], [12]. The graphic model representation of the latent class model is shown in Figure 3.

B. Series-Deadline Model

Conferences are often denoted by series name and year. In the expression "RecSys 2015", the conference series name is "RecSys", and the year of the event is 2015. It is natural to assume that a researcher who is interested in RecSys 2014 will be interested in RecSys 2015. Therefore we can assume that a researcher would be equally interested in each one of the conferences of the same series.

The time of the event is also important, a researcher tends to pay attention to interested conference in one's active period: a PhD student enrolled in 2014 might track RecSys 2014, but she/he is unlikely to track RecSys 2012, which is outdated.

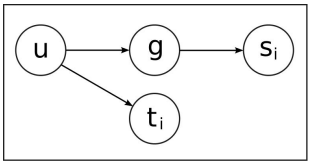

Fig. 4. Graphical model representation of series-deadline model.

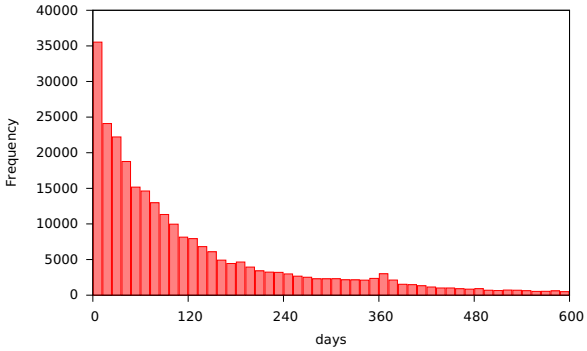

Fig. 5. Distribution of the time difference between a pair of CFPs tracked by the same user.

Therefore we propose a model that describe the probability that a researcher u will track a conference CFP i by combining both the interest to the series of the conference and the impact of time by simply multiplying them:

$$P(i|u) = P(s_i|u)P(t_i|u) \tag{10}$$

where $P(s_i|u)$ is the probability that a researcher u might be interested with the series of the conference and $P(t_i|u)$ is the probability that a researcher might consider the time of the conference appropriate. The graphical model representation of series-deadline model is presented in Figure 4.

We model the probability $P(s_i|u)$ that describes the interest of the user u by latent class model:

$$P(s_i|u) = \sum_{g \in G} P(s_i|g)P(g|u) \tag{11}$$

which assigns people to different latent interest groups and estimates their preferences by adding up the preferences of the groups they belong to.

The second part $P(t_i|u)$ is a probability that describe whether the time of the conference is appropriate for the user. The time of the conference can be inappropriate for various reasons. Paper submission may be closed before the user ever sees its CFP. Time of the conference may be too late to meet the user's plan. A conference has many time-related attributes: there are the opening date, closing date, submission deadline, notification due, and final version due. Among these attributes, we believe that submission deadline is the most concerned time by a researcher. Therefore, we use the submission deadline as the time of the conference.

We assume that the conferences whose CFPs are tracked by the same user should occur in temporal proximity. This means

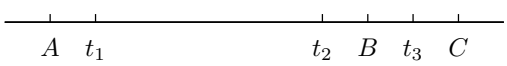

Fig. 6. A user tracks CFPs whose deadlines are t_1, t_2, t_3, and t_4. But the system only knows three of them. Where should t_4 be: A, B, or C?

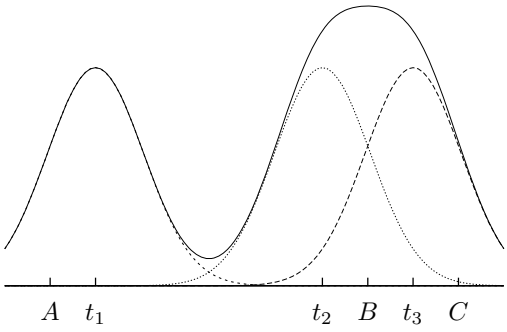

Fig. 7. A Gaussian mixture model in which the deadline of every CFP tracked by the user is used as the center of a component.

a CFP whose deadline is close to the CFPs already tracked by a user would be more likely to get tracked by the user than other CFPs.

To validate our assumption, we randomly select several pairs of CFPs in which each pair of them are tracked by the same user, and show the distribution of the pair-wise time difference (of submission deadlines) in a histogram (Fig. 5). We can see that CFPs tracked by the same user tend to be temporal proximate to each other. Most intervals between two deadlines are within 120 days or four months.

Also, a user might track several CFPs that have different submission deadlines. And the distribution of these deadlines might not have a unified form, since every user's tracked CFPs might be different from one another's.

A case study shows that a reasonable distribution should be dependent of the deadline of every CFP the user tracks. As shown in Fig. 6, the system knowns the user tracks three CFPs whose deadlines are t_1, t_2, and t_3, but the deadline of another CFP tracked by the user is unknown. A most reasonable guess is that the unknown deadline should be at the position of B, since it is proximate to both t_2 and t_3. The position of C is less likely, since it is only proximate to t_3 but is a little far away from t_2. A is the least likely one of the three candidate position, since it is far away from t_2 and t_3.

We model this temporal proximity by a Gaussian mixture, as shown in Fig. 7. Every component in the mixture corresponds to a CFP tracked by the user and is centered at the deadline of the corresponding CFP. If a user u_i tracks CFPs whose deadlines are $t_{i1}, t_{i2}, ..., t_{im_i}$, then the probability that the user u_i tracks a CFP whose deadline is t is:

$$P_{\sigma^2}(t|u_i) = \frac{1}{m_i}\sum_{j=1}^{m_i} N_{t_{ij},\sigma^2}(t) \tag{12}$$

where every component shares the same height and standard deviation:

$$N_{\mu,\sigma^2}(t) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left\{-\frac{(t-\mu)^2}{2\sigma^2}\right\} \tag{13}$$

If we have other training samples $\hat{t}_{i1}, \hat{t}_{i2}, ..., \hat{t}_{ik_i}$, we can compute the total log likelihood:

$$logL = \sum_{i=1}^{N}\sum_{j=1}^{k_i} logP_{\sigma^2}(\hat{t}_{ij}|u_i) \tag{14}$$

To maximize the log likelihood, we can derive an iterative update equation for the squared deviation:

$$\sigma^2_{new} = \frac{1}{\sum_{i=1}^{N} k_i}\sum_{i=1}^{N}\sum_{k=1}^{k_i} \frac{\sum_{j=1}^{m_i}(\hat{t}_{ik} - t_{ij})^2 N_{t_{ij},\sigma^2}(\hat{t}_{ik})}{\sum_{j=1}^{m_i} N_{t_{ij},\sigma^2}(\hat{t}_{ik})} \tag{15}$$

Since we use the paper submission deadline as the time of the conference, we name this proposed model the series-deadline model.

V. Experiments

We use *tracked-by* relations extracted from open-access data on WikiCFP as our major data set. We also crawled a conference series list from WikiCFP to build links between a CFP to its correspondent conference series. For CFPs that can not link to a series in the list, we extract the series names from the summaries of the CFPs and treat all CFPs with the same series name as from the same series. For example, for a conference whose summary is "RecSys'13", our extractor based on heuristics will extract "RecSys" as the series name. The submission deadline is directly extracted from the xml data on WikiCFP.

We carry on a five-fold cross-validation in our experiment in order to eliminate the bias in test data selection. In this evaluation scenario, the data set is randomly split into five chunks. We run the experiment five times. Each time the union of four chunks serves as the train set, i.e, the feedbacks that is already known by the recommender system, and the rest one serves as the test set, which is compared to the recommended items to evaluate the performance of the system. Performance score of cross-validation is the average score of individual runs.

When evaluating the recommended items, CFPs that are tracked by the currently evaluated user in the test set are treated relevant, others irrelevant.

In order to evaluate the effects of collaborative filtering techniques on CFP recommendation, we adopt two different metrics: $RPrecision$ [17], and normalized Discounted Cumulative Gain ($nDCG$) [18].

$RPrecision$ is the precision when the system recommends *R* items, in which *R* is the number of items in the test set that is relevant to the currently evaluated user. It should be noted that the recall equals the precision when recommending R items. [17] shows that $RPrecision$ is very stable in comparing

different retrieval techniques. $RPrecision$ is computed with the following equation:

$$RPrecision = \frac{\sum_{i=1}^{R} rel_i}{R} \tag{16}$$

in which rel_i is the relevance of the *i*th recommended item. $rel_i = 1$ if the item is relevant to the evaluated user, and $rel_i = 0$ if otherwise.

Normalized Discounted Cumulative Gain ($nDCG$ or normalized DCG) takes not only the relevance of recommended item into account, but also stresses the importance of the ranked position of these relevant items. DCG gives relevant items that ranked high more credit by applying a discounted factor that reduces the gain from relevant items at lower position. An idealized DCG (iDCG) is the DCG score under the condition that we precisely rank items in descending order of their relevance scores. The magnitude of DCG scores varies with data, as [18] notes "it is difficult to assess the magnitude of the difference of two (D)CG curves". Therefore, [18] propose a normalized version of DCG that divide DCG by the idealized DCG.

$$DCG@N = \sum_{i=1}^{N} \frac{rel_i}{log_2(i+1)} \tag{17}$$

$$iDCG@N = \sum_{i=1}^{min(N,R)} \frac{1}{log_2(i+1)} \tag{18}$$

$$nDCG@N = \frac{DCG@N}{iDCG@N} \tag{19}$$

VI. Results and Discussions

Table I shows the performance of different collaborative filtering methods. For the neighborhood and neighborhood-idf methods, the number of selected neighbors is 15. For latent class model, number of latent class is 400. For series model and series deadline model, number of latent class is 100. The parameters are tuned by manually search over the parameter space. We only compare $nDCG$ at *N*=20 for brevity since we find the rankings of different methods do not change under both smaller and larger *N*.

Comparing the result, we can see popularity, as we anticipated, performs badly. We attribute this result to the long-tailed nature of the data. Also, the modification to Jaccard index does increase the effectiveness of neighborhood method as we expected. It also turns out the latent class model is superior to other baselines. The proposed series-deadline model outperforms all other methods.

We perform a paired t-test on every pair of methods. A paired t-test is used when comparing two sets of data where both sets are from the same set of test subjects but are outputs from different procedures. In our experiment, the collaborative filtering algorithms are tested on the same set of user. The $nDCG$ and $RPrecision$ measurement on the data set is the mean value of the measurements for all single users in the data set (Eq. 20-21). We compare the sets of per user measurements with a paired t-test. We found that between methods "neighborhood" and "neighborhood-idf" the $nDCG$ difference is 95% significant and the $RPrecision$ difference is not significant. The differences between all the other pairs of methods are 99% significant. In conclusion, the proposed series-deadline model statistically significantly outperforms the latent factor model on WikiCFP data set, while the effectiveness of inverse document frequency in Jaccard index is not conclusive.

TABLE I. Comparison of different collaborative filtering algorithms. The proposed series-deadline model has the best overall performance.

Method	$RPrecision$	$nDCG@20$
popularity	0.0046	0.0203
neighborhood	0.0302	0.1252
neighborhood-idf	0.0309	0.1286
latent class model	0.0354	0.1382
series-deadline model	0.0373	0.1460

$$nDCG = \sum_{u} nDCG_u \tag{20}$$

$$RPrecision = \sum_{u} RPrecision_u \tag{21}$$

We also explore the impact that the value of parameters has on the performance of recommender systems. For neighborhood-based methods, we evaluate the $RPrecision$ and $nDCG@20$ of neighborhood and neighborhood-idf algoritms under several selected neighborhood sizes. It can be observed neighborhood size that neither too small nor too large will generate positive results.

The explanation for this is that neighborhood too small will produce too few recommendations and decrease recall. Whereas, increasing neighborhood will include more like-minded users into the neighborhood and increase precision. On the other hand, too large neighborhood will include noisy users into the neighborhood, deteriorate the quality of recommendations, and decrease recall in the end.

For latent class model and series-deadline model, we explore the relationship between the performance and the number of latent classes, as shown in Fig. 9. A latent class represents a virtual community that best describes a mode of user behaviors on the entire set. Ideally, the more latent classes, the more accurately the model can describe the data set. However, adding more latent classes brings slower convergent speed, which in turn undermines training quality of the model at a limited budget of time. More latent classes also makes the model more complex and more likely to end up in overfitting.

The time complexity of latent class model is analysed in [19]. A single EM iteration takes $O(kN)$ time to compute where k denotes the number of latent classes and N denotes the total number of tracking relations. The time complexity of series-deadline model is dominated by the part that models the interest of a user to a conference series, which itself is a latent class model. In our experiment, the series-deadline model takes shorter time to train compared to the latent class model. A straightforward explanation is that the optimal latent class number of the series-deadline model in our experiment is smaller than that of the latent class model (Fig. 9).

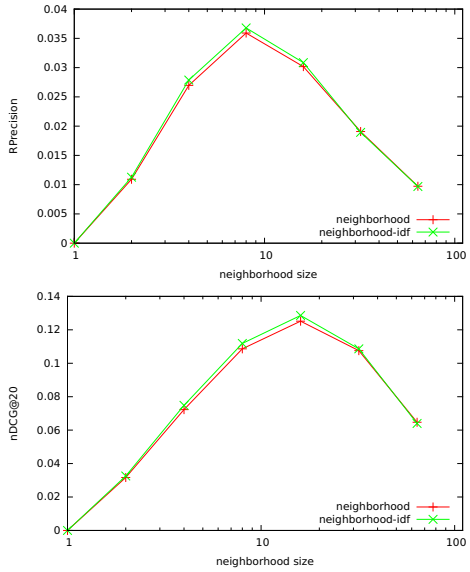

Fig. 8. The performance of neighborhood-based collaborative filtering algorithms is closely related to neighborhood size. Neighborhood that either too small or too big will undermine the performance of neighborhood-based CF. It can be seen that the refinement to Jaccard index increase both $RPrecision$ and $nDCG$.

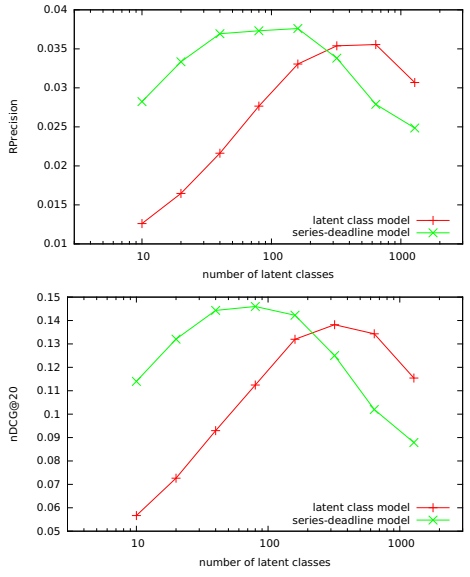

Fig. 9. Performance vs. number of latent classes. The optimal latent class number of series-deadline model is smaller than the latent class model since the number of series is less than that of CFPs.

VII. Conclusion

We have employed several most-used collaborative filtering approaches to recommend call for papers (CFPs) to researchers. We find that popularity, a non-personalized method, has poor performance, which indicates that the users of WikiCFP have diverse interests. We find that a proper weighting to Jaccard similarity improves the performance of neighborhood-based approaches. We have explored the impact of parameter to the effectiveness of collaborative filtering methods. Comparing the baselines in our experiment, we find that latent class model has better performance over the other baselines.

We also propose a hybrid approach that melts series and time of conference into collaborative filtering. Experiment shows that our method outperforms all other approaches.

The result of our hybrid approach shows that combining knowledge with collaborative filtering is promising for CFP recommendation. Though, several aspects about CFP recommendation remain unexplored. CFPs are documents with highly topic-oriented texts, which means combining content information with collaborative filtering may produce better recommendations. CFPs are also linked data: one typical CFP is associated with an academic event, which in turn is linked with the accepted papers in the past. Since integrating ontology into system can enhance the performance of CFP retrieval. It is also reasonable to assume that utilizing linked knowledge will help us to recommend relevant CFPs better. Other attributes of CFPs, such as when and where the event will be held and folksonomy categories, are also useful for building a more effective recommender system.

References

[1] T. Huynh, A. Takasu, T. Masada, and K. Hoang, "Collaborator recommendation for isolated researchers," in *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on*, May 2014, pp. 639–644.

[2] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '11. New York, NY, USA: ACM, 2011, pp. 448–456. [Online]. Available: http://doi.acm.org/10.1145/2020408.2020480

[3] G. Hurtado Martín, S. Schockaert, C. Cornelis, and H. Naessens, "An Exploratory Study on Content-Based Filtering of Call for Papers," in *Multidisciplinary Information Retrieval*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, vol. 8201, pp. 58–69. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-41057-4_7

[4] J. Xia, K. Wen, R. Li, and X. Gu, "Optimizing academic conference classification using social tags," in *Computational Science and Engineering (CSE), 2010 IEEE 13th International Conference on*, Dec 2010, pp. 289–294.

[5] L. Issertial and H. Tsuji, "Data management and user interface for a call for paper manager," in *Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual*, July 2013, pp. 463–466.

[6] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 43–52. [Online]. Available: http://dl.acm.org/citation.cfm?id=2074094.2074100

[7] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews," in *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, ser. CSCW '94. New

York, NY, USA: ACM, 1994, pp. 175–186. [Online]. Available: http://doi.acm.org/10.1145/192844.192905

[8] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *Internet Computing, IEEE*, vol. 7, no. 1, pp. 76–80, Jan 2003.

[9] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 143–177, Jan. 2004. [Online]. Available: http://doi.acm.org/10.1145/963770.963776

[10] M. Gori and A. Pucci, "Itemrank: A random-walk based scoring algorithm for recommender engines," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, ser. IJCAI'07. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007, pp. 2766–2771. [Online]. Available: http://dl.acm.org/citation.cfm?id=1625275.1625720

[11] H. Yildirim and M. S. Krishnamoorthy, "A random walk method for alleviating the sparsity problem in collaborative filtering," in *Proceedings of the 2008 ACM Conference on Recommender Systems*, ser. RecSys '08. New York, NY, USA: ACM, 2008, pp. 131–138. [Online]. Available: http://doi.acm.org/10.1145/1454008.1454031

[12] T. Hofmann and J. Puzicha, "Latent class models for collaborative filtering," in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, ser. IJCAI '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 688–693. [Online]. Available: http://dl.acm.org/citation.cfm?id=646307.687583

[13] P. Jaccard, "Etude comparative de la distribution florale dans une portion des alpes et du jura," *Bulletin de la Société Vaudoise des Sciences Naturelles*, vol. 37, pp. 547–579, 1901.

[14] D. J. Rogers and T. T. Tanimoto, "A computer program for classifying plants," *Science*, vol. 132, no. 3434, pp. 1115–1118, 1960. [Online]. Available: http://www.sciencemag.org/content/132/3434/1115.short

[15] K. Sparck Jones, "Document retrieval systems," P. Willett, Ed. London, UK, UK: Taylor Graham Publishing, 1988, ch. A Statistical Interpretation of Term Specificity and Its Application in Retrieval, pp. 132–142. [Online]. Available: http://dl.acm.org/citation.cfm?id=106765.106782

[16] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977. [Online]. Available: http://web.mit.edu/6.435/www/Dempster77.pdf

[17] C. Buckley and E. M. Voorhees, "Evaluating evaluation measure stability," in *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '00. New York, NY, USA: ACM, 2000, pp. 33–40. [Online]. Available: http://doi.acm.org/10.1145/345508.345543

[18] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, Oct. 2002. [Online]. Available: http://doi.acm.org/10.1145/582415.582418

[19] T. Hofmann, "Collaborative filtering via gaussian probabilistic latent semantic analysis," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, ser. SIGIR '03. New York, NY, USA: ACM, 2003, pp. 259–266. [Online]. Available: http://doi.acm.org/10.1145/860435.860483